# Process Scheduling

## First Come First Serve:

### Code:

```
#include<iostream>
using namespace std;
int main(){
	int at[10],bt[10],wt[10],tat[10],n,burst,complete,temp1,temp2,p[10],temp3,comp;
	float avg_tat,avg_wt,TAT,WT;
	cout<<"Enter the no. of proess:"<<endl;
	cin>>n;
	for(int i=0;i<n;i++){
		cout<<"Enter the arrival time and burst time of process"<<i+1<<" ";
		cin>>at[i];
		cin>>bt[i];
		p[i]=i+1;
	}
	for(int i=0;i<n;i++){
		for(int j=0;j<n;j++){
			if(at[i]<at[j]){
				temp1=at[i];
				at[i]=at[j];
				at[j]=temp1;
				temp2=bt[i];
				bt[i]=bt[j];
				bt[j]=temp2;
				temp3=p[i];
				p[i]=p[j];
				p[j]=temp3;
			}
		}
	}
	cout<<"arranged order of process according to arrival time:"<<endl;
	for(int i=0;i<n;i++){
		cout<<"process"<<p[i]<<" "<<at[i]<<" "<<bt[i]<<endl;
	}
	complete=0;
	for(int i=0;i<n;i++){
			if(at[i]==0){
				wt[i]=at[i];
				complete=bt[i];
				tat[i]=complete-at[i];
			}
```

```
				else if(at[i]>complete){
					complete++;
					i=i-1;
				}
			else{
				complete+=bt[i];
				tat[i]=complete-at[i];
					wt[i]=tat[i]-bt[i];
				}
		}
		for(int i=0;i<n;i++){
			TAT+=tat[i];
			WT+=wt[i];
		}
		avg_tat=TAT/n;
		avg_wt=WT/n;
		cout<<"process\t"<<"waiting\t"<<"tat\t"<<endl;
		for(int i=0;i<n;i++){
			cout<<p[i]<<"\t"<<wt[i]<<"\t"<<tat[i]<<endl;
		}
		//cout<<TAT<<" "<<WT<<endl;
		cout<<"Average turnaraound time is: "<<avg_tat<<endl;
		cout<<"Average waiting time is: "<<avg_wt<<endl;
}
```

## Shortest Job First:

Code:

```
#include<iostream>
using namespace std;

int main(){
		int at[10],bt[10],wt[10],tat[10],n,burst,complete,temp,p[10];
		float avg_tat,avg_wt,TAT,WT;
		cout<<"Enter the no. of process:"<<endl;
		cin>>n;
		for(int i=0;i<n;i++){
			cout<<"Enter the arrival time and burst time of process"<<i+1<<" ";
			cin>>at[i];
			cin>>bt[i];
			p[i]=i+1;
		}

		complete=0;
//		int temp2=0;
```

```
for(int i=0;i<n;i++){
                if(i==0){
                        wt[i]=at[i];
                        complete+=bt[i];
                        tat[i]=complete-at[i];
                }
                else{

                        for(int j=1;j<n;j++){
                                if(bt[i]<bt[j]){
                                        temp=bt[i];
                                        bt[i]=bt[j];
                                        bt[j]=temp;
                                        temp=at[i];
                                        at[i]=at[j];
                                        at[j]=temp;
                                        temp=p[i];
                                        p[i]=p[j];
                                        p[j]=temp;
                                        //temp2=1;
                                }
                        }
                }
        }
        for(int i=1;i<n;i++){
                complete+=bt[i];
                tat[i]=complete-at[i];
                wt[i]=tat[i]-bt[i];
        }
/*      cout<<"arranged order of process according to arrival time:"<<endl;
for(int i=0;i<n;i++){
        cout<<"process"<<p[i]<<" "<<at[i]<<" "<<bt[i]<<" "<<complete<<endl;
}       */
for(int i=0;i<n;i++){
        TAT+=tat[i];
        WT+=wt[i];
}
avg_tat=TAT/n;
avg_wt=WT/n;
cout<<"process\t arrival time\t"<<"burst time\t waiting time\t"<<"turnaround time"<<endl;
for(int i=0;i<n;i++){
        cout<<p[i]<<"\t\t"<<at[i]<<"\t\t"<<bt[i]<<"\t\t"<<wt[i]<<"\t\t"<<tat[i]<<endl;
}
cout<<"Average turnaraound time is: "<<avg_tat<<endl;
```

```
	cout<<"Average waiting time is: "<<avg_wt<<endl;
}
```

## Shortest Remaining Job First:

**Code:**

```
#include<iostream>
using namespace std;
int main()
{
	int at[10], bt[10], x[10];
	int waiting[10], turnaround[10], completion[10];
	int i, j, smallest, count = 0, time, n;
	double avg = 0, tt = 0, end;
	cout<<"Enter the number of processes:"<<endl;
	cin>>n;
	for(int i=0;i<n;i++){
		cout<<"Enter the arrival time and burst time of process"<<i+1<<" ";
		cin>>at[i];
		cin>>bt[i];
	}
	for(i=0;i<n;i++)
	{
		x[i] = bt[i];
	}
		bt[9] = 999;


	time = 0;
	while(count!= n)
	{

		smallest = 9;
		for(i=0;i<n;i++)
		{
			if (at[i]<=time && bt[i] < bt[smallest] && bt[i] > 0)
				smallest = i;
		}

		bt[smallest] --;
		if (bt[smallest] == 0)
		{
			count ++;
			end = time+1;
```

```
                        completion[smallest] = end;
                        waiting[smallest] = end - at[smallest] - x [smallest];
                        turnaround[smallest] = end - at[smallest];

                }
                time++;
        }

        cout<<"Process"<<"\t"<<"arrival time"<<"\t"<<"burst time"<<"\t"<<"Waiting
time"<<"\t"<<"turrnaound time"<<endl;
        for(i=0;i<n;i++)
        {

        cout<<"P"<<i+1<<"\t\t"<<at[i]<<"\t\t"<<x[i]<<"\t\t"<<waiting[i]<<"\t\t"<<turnaround[i]<<endl;
                avg += waiting[i];
                tt += turnaround[i];
        }
        cout<<"\n Average waiting time = "<<avg/n<<"\n";
        cout<<"Average turanound time = "<<tt/n<<endl;
        return 0;
        }
```

## Round-Robin:

Code:

```
#include <iostream>
using namespace std;
// #include<conio.h>

int main()
{
  // initlialize the variable name
  int i, NOP, sum = 0, count = 0, y, quant, wt = 0, tat = 0, at[10], bt[10], temp[10];
  float avg_wt, avg_tat;
  cout<<" Total number of process in the system: ";
  cin>>NOP;
  y = NOP; // Assign the number of process to variable y

  // Use for loop to enter the details of the process like Arrival time and the Burst Time
  for (i = 0; i < NOP; i++)
  {
    cout<<"\n Enter the Arrival and Burst time of the Process"<< i + 1<<":";
    cin>>at[i];
    cin>>bt[i];
    temp[i] = bt[i]; // store the burst time in temp array
  }
  // Accept the Time qunat
  cout<<"Enter the Time Quantum for the process: ";
  cin>>quant;
  // Display the process No, burst time, Turn Around Time and the waiting time
  cout<<"\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ";
  for (sum = 0, i = 0; y != 0;)
  {
    if (temp[i] <= quant && temp[i] > 0) // define the conditions
    {
        sum = sum + temp[i];
        temp[i] = 0;
        count = 1;
    }
    else if (temp[i] > 0)
    {
        temp[i] = temp[i] - quant;
        sum = sum + quant;
    }
    if (temp[i] == 0 && count == 1)
    {
        y--; // decrement the process no.
        cout<<"\nProcess No["<<i+1<<"] \t\t "<<bt[i]<<"\t\t\t\t "<<sum-at[i]<<"\t\t\t "<<sum - at[i] -
bt[i];
        wt = wt + sum - at[i] - bt[i];
        tat = tat + sum - at[i];
        count = 0;
    }
    if (i == NOP - 1)
    {
        i = 0;
    }
    else if (at[i + 1] <= sum)
    {
        i++;
    }
    else
    {
        i = 0;
    }
  }
  // represents the average waiting time and Turn Around time
  avg_wt = wt * 1.0 / NOP;
  avg_tat = tat * 1.0 / NOP;
  cout<<"\n Average Turn Around Time: "<<avg_wt;
  cout<<"\n Average Waiting Time: "<<avg_tat;
}
```

## Priority Scheduling:

Code:

```
#include<iostream>

using namespace std;

int main()
{
    int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
    cout<<"Enter Total Number of Process:";
    cin>>n;

    cout<<"\nEnter Burst Time and Priority\n";
    for(i=0;i<n;i++)
    {
        cout<<"\nP["<<i+1<<"]\n";
        cout<<"Burst Time:";
        cin>>bt[i];
        cout<<"Priority:";
        cin>>pr[i];
        p[i]=i+1;           //contains process number
    }

    //sorting burst time, priority and process number in ascending order using selection sort
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(pr[j]<pr[pos])
                pos=j;
        }

        temp=pr[i];
        pr[i]=pr[pos];
        pr[pos]=temp;

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;            //waiting time for first process is zero

    //calculate waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=total/n;      //average waiting time
    total=0;

    cout<<"\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time";
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        total+=tat[i];
        cout<<"\nP["<<p[i]<<"]\t\t  "<<bt[i]<<"\t\t    "<<wt[i]<<"\t\t\t"<<tat[i];
    }

    avg_tat=total/n;     //average turnaround time
    cout<<"\n\nAverage Waiting Time="<<avg_wt;
    cout<<"\nAverage Turnaround Time="<<avg_tat;

    return 0;
}
```

## Page Replacement Algorithm

### First In First Out:

**Code:**

```
#include <bits/stdc++.h>
```

```
using namespace std;

const int N=100;

void fifo_page_replacement(int frame_size, int n,int pages[])
{
            int mark[N];     queue<int> Q;
   int page_faults=0;

   for(int i=0; i<n; i++)
   {
      if(mark[pages[i]]==true)
      {
      }        else
      {
         Q.push(pages[i]);
                                 mark[pages[i]]=true;
         if(Q.size()>frame_size)
         {
            int p= Q.front();
                                             mark[p]=false;
            Q.pop();
         }
         page_faults++;

      }

   }
   cout<<"Frame size: "<<frame_size<<endl;
            cout<<"Page faults: "<<page_faults<<endl;
            cout<<"Page Hits: "<<n-page_faults<<endl;
            return;
}
int main()
{
            int frame_size=4; int pages[N];
int n;
cout<<"Enter the frame size:"; cin>>frame_size;
   cout<<"Page Reference Stream Length: ";     cin>>n;
 cout<<"Page Reference Stream:\n";     for(int i=0; i<n; i++)        cin>>pages[i];
            fifo_page_replacement(frame_size,n,pages);      return 0;
}
```

## Second Chance Page Replacement

Code:

```
#include<iostream>
 using namespace std;
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;
int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
}
 int getHitIndex(int data)
{
    int hitind;
    for(k=0; k<nf; k++)
    {
        if(p[k]==data)
        {
            hitind=k;
            break;
        }
    }
    return hitind;
}
void secondchance()
{
    int usedbit[50];
    int victimptr=0;
pgfaultcnt=0;
    for(i=0; i<nf; i++)
```

```
   p[i]=9999;    for(i=0; i<nf; i++)
   usedbit[i]=0;
for(i=0; i<n; i++)
{
   cout<<"\nFor "<<in[i]<<" :";
   if(isHit(in[i]))
   {
      cout<<"No page fault!";
      int hitindex=getHitIndex(in[i]);
      if(usedbit[hitindex]==0)
         usedbit[hitindex]=1;
   }
   else
   {
      pgfaultcnt++;
      if(usedbit[victimptr]==1)
      {
         do
         {
            usedbit[victimptr]=0;
            victimptr++;
            if(victimptr==nf)
               victimptr=0;
         }
         while(usedbit[victimptr]!=0);
      }
      if(usedbit[victimptr]==0)
      {
         p[victimptr]=in[i];
         usedbit[victimptr]=1;
         victimptr++;
      }

for (k=0; k<nf; k++)
{
   if(p[k]!=9999)
      cout<<p[k];
}

   }
   if(victimptr==nf)
      victimptr=0;
}
cout<<"\nTotal no of page faults: "<<pgfaultcnt;
```

```
}


int main()
{
    cout<<"\nEnter length of page reference sequence:";
    cin>>n;
    cout<<"\nEnter the page reference sequence:";
    for(i=0; i<n; i++)
        cin>>in[i];
    cout<<"\nEnter no of frames:";
    cin>>nf;
          // fifo();
          // optimal();
           //lru();
           //lfu();
          secondchance();
}
```

## Optimal page replacement

Code:

```
#include<iostream>
 using namespace std;
int n,nf;
int in[100],p[50],hit=0;
int i,j,k,pgfaultcnt=0;
int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
}

void optimal()
{
```

```
pgfaultcnt=0;
for(i=0; i<nf; i++)
   p[i]=9999;
int near[50];
for(i=0; i<n; i++)
{

   cout<<"\nFor "<<in[i]<<": ";

   if(isHit(in[i])==0)
   {

      for(j=0; j<nf; j++)
      {
         int pg=p[j];
         int found=0;
         for(k=i; k<n; k++)
         {
            if(pg==in[k])
            {
               near[j]=k;
               found=1;
               break;
            }
            else
               found=0;
         }
         if(!found)
            near[j]=9999;
      }
      int max=-9999;
      int repindex;
      for(j=0; j<nf; j++)
      {
         if(near[j]>max)
         {
            max=near[j];
            repindex=j;
         }
      }
      p[repindex]=in[i];
      pgfaultcnt++;

      for (k=0; k<nf; k++)
```

```
    {
        if(p[k]!=9999)
            cout<<p[k];
    }

        }
        else
            cout<<"No page fault";
    }
        cout<<"\nTotal no of page faults:"<<pgfaultcnt);

}

int main()
{
    cout<<"\nEnter length of page reference sequence:";
    cin>>n;
    cout<<"\nEnter the page reference sequence:";
    for(i=0; i<n; i++)
        cin>>in[i];
    cout<<"\nEnter no of frames:";
    cin>>nf;
    optimal();
}
```

## LRU page replacement

Code:

```
#include<iostream>
 using namespace std;
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;
int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
}
void lru()
{
pgfaultcnt=0;
    for(i=0; i<nf; i++)
        p[i]=9999;
    int least[50];
    for(i=0; i<n; i++)
    {

        cout<<"\nFor "<<in[i]<<":";

        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
            {
                int pg=p[j];
                int found=0;
                for(k=i-1; k>=0; k--)
                {
                    if(pg==in[k])
                    {
                        least[j]=k;
                        found=1;
                        break;
                    }
                    else
                        found=0;
                }
                if(!found)
                    least[j]=-9999;
            }
            int min=9999;
            int repindex;
            for(j=0; j<nf; j++)
            {
                if(least[j]<min)
                {
                   min=least[j];
                   repindex=j;
               }
            }
            p[repindex]=in[i];
            pgfaultcnt++;

for (k=0; k<nf; k++)
    {
        if(p[k]!=9999)
            cout<<p[k];
    }       }
        else
            cout<<"No page fault!";
    }
              cout<<"\nTotal no of page faults: "<<pgfaultcnt;

}


int main()
{
    cout<<"\nEnter length of page reference sequence:";
    cin>>n;
    cout<<"\nEnter the page reference sequence:";
    for(i=0; i<n; i++)
        cin>>in[i];
    cout<<"\nEnter no of frames:";
    cin>>nf;
            lru();
}
```

## LFU page replacement

Code:

```
#include<iostream>
 using namespace std;
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;
int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
}
 int getHitIndex(int data)
{
    int hitind;
    for(k=0; k<nf; k++)
    {
        if(p[k]==data)
        {
            hitind=k;
            break;
        }
    }
    return hitind;
}

void lfu()
{
    int usedcnt[100];
    int least,repin,sofarcnt=0,bn;
pgfaultcnt=0;
    for(i=0; i<nf; i++)
        p[i]=9999;
    for(i=0; i<nf; i++)
        usedcnt[i]=0;

    for(i=0; i<n; i++)
    {

        Cout<<"\n For "<<in[i]<<" : ";
        if(isHit(in[i]))
        {
            int hitind=getHitIndex(in[i]);
            usedcnt[hitind]++;
            cout<<"No page fault!";
        }
        else
        {
            pgfaultcnt++;
            if(bn<nf)
            {
                p[bn]=in[i];
                usedcnt[bn]=usedcnt[bn]+1;
                bn++;
            }
            else
            {
                least=9999;
                for(k=0; k<nf; k++)
                    if(usedcnt[k]<least)
                    {
                        least=usedcnt[k];
                        repin=k;
                    }
                p[repin]=in[i];
                sofarcnt=0;
                for(k=0; k<=i; k++)
                    if(in[i]==in[k])
                        sofarcnt=sofarcnt+1;
                usedcnt[repin]=sofarcnt;
            }

 for (k=0; k<nf; k++)
    {
        if(p[k]!=9999)
            cout<<p[k];
    }
         }

    }
    cout<<"\nTotal no of page faults: "<<pgfaultcnt;
}

int main()
{
    cout<<"\nEnter length of page reference sequence:";
    cin>>n;
    cout<<"\nEnter the page reference sequence:";
    for(i=0; i<n; i++)
        cin>>in[i];
    cout<<"\nEnter no of frames:";
    cin>>nf;
            lfu();
}
```

## Disk Scheduling

### First Come First Serve

#### Code

```
#include<iostream>
#include<math.h>
using namespace std;
int main(){

    int i,j,k,n,m,sum=0,x,y,h;
    cout<<"*** FCFS Disk Scheduling Algorithm ***\n";

    cout<<"Enter the size of disk\n";
    cin>>m;
    cout<<"Enter number of requests\n";
    cin>>n;
    cout<<"Enter the requests\n";

    // creating an array of size n
    int a[n];
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    for(i=0;i<n;i++){
        if(a[i]>m){
            cout<<"Error, Unknown position "<<a[i]<<"\n";
            return 0;
        }
    }
    cout<<"Enter the head position\n";
    cin>>h;

    // head will be at h at the starting
    int temp=h;
    cout<<temp;
    for(i=0;i<n;i++){
        cout<<" -> "<<a[i]<<' ';
        // calculating the difference for the head movement
        sum+=abs(a[i]-temp);
        // head is now at the next I/O request
        temp=a[i];
    }
    cout<<'\n';
    cout<<"Total head movements = "<< sum<<'\n';
    return 0;
}
```

## Shortest Seek time First

Code:

```
#include<iostream>
//#include<conio.h>
#include<math.h>
using namespace std;
int main()
{
        int queue[100],t[100],head,seek=0,n,i,j,temp;
      // clrscr();
        cout<<"*** SSTF Disk Scheduling Algorithm ***\n";
        cout<<"Enter the size of Queue\t";
        cin>>n;
        cout<<"Enter the Queue\t";
        for(i=0;i<n;i++)
        {
                scanf("%d",&queue[i]);
        }
        cout<<"Enter the initial head position\t";
        cin>>head;
        for(i=1;i<n;i++)
        t[i]=abs(head-queue[i]);
        for(i=0;i<n;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(t[i]>t[j])
                        {
                                temp=t[i];
                                t[i]=t[j];
                                t[j]=temp;
                                temp=queue[i];
```

```
                                        queue[i]=queue[j];
                                        queue[j]=temp;
                                }
                        }
                }
                for(i=1;i<n-1;i++)
                {
                        seek=seek+abs(head-queue[i]);
                        head=queue[i];
                }
                cout<<"\nTotal Seek Time is "<<seek;
}
```

## SCAN

Code:

```
#include <iostream>
#include<math.h>

using namespace std;

int main()
{
 //  clrscr();
   int n,d[8],a,b,c=0,j,i=0;
   char ch='y';
   cout<<"Enter no. of request : ";
   cin>>n;
   cout<<"enter value of initial head position : ";
   cin>>a;


   cout<<"Enter the requests ";

   for(i=0;i<n;i++)
   {
      cin>>d[i];
   }

   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {

         if(d[i]<d[j])
```

```
            {
                b=d[i];
                d[i]=d[j];
                d[j]=b;
            }

        }
    }

    for(i=0;i<n;i++)
    {
        if(d[i]>a)
        {
            j=i;
            break;
        }
    }

    c=0;
    b=0;

    do
    {
        c+=abs(b-d[j]);
        b=d[j];
        j++;
    }while(j<n);

    c=c+a;
    cout<<" \nTotal head movement = "<<c<<" cylinders";
}
```

## C-SCAN

Code:

```
#include <iostream.h>

#include <math.h>

int main()

{

    int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],
    queue2[20], temp1 = 0, temp2 = 0;
    cout<<"Enter the max range of disk\n";
```

```
cin<<max;

cout<<"Enter the initial head position\n";

cin>>head;

cout<<"Enter the size of queue request\n";

cin>>n;

cout<<"Enter the queue of disk positions to be read\n";

for (i = 1; i <= n; i++)

{

  cin>>temp;

  if (temp >= head)

  {

    queue1[temp1] = temp;

    temp1++;
  }

  else

  {

    queue2[temp2] = temp;

    temp2++;
  }
}

for (i = 0; i < temp1 - 1; i++)

{

  for (j = i + 1; j < temp1; j++)
```

```
    {

        if (queue1[i] > queue1[j])

        {

            temp = queue1[i];

            queue1[i] = queue1[j];

            queue1[j] = temp;
        }
    }
}

for (i = 0; i < temp2 - 1; i++)

{

    for (j = i + 1; j < temp2; j++)

    {

        if (queue2[i] > queue2[j])

        {

            temp = queue2[i];

            queue2[i] = queue2[j];

            queue2[j] = temp;
        }
    }
}

for (i = 1, j = 0; j < temp1; i++, j++)

    queue[i] = queue1[j];

queue[i] = max;

queue[i + 1] = 0;
```

```
    for (i = temp1 + 3, j = 0; j < temp2; i++, j++)

        queue[i] = queue2[j];

    queue[0] = head;

    for (j = 0; j <= n + 1; j++)

    {

        diff = abs(queue[j + 1] - queue[j]);

        seek += diff;

        cout<<"Disk head moves from "<<queue[j]<<" to "<<queue[j+1]<<" with seek "<<diff<<endl;
    }

    cout<<"Total seek time is "<<seek<<endl;
    return 0;
}
```

LOOK

Code:

```
#include <iostream>

#include <stdlib.h>


#define LOW 0

#define HIGH 199

using namespace std;


int main(){

  int queue[20], head, q_size, i,j, seek=0, diff, max, temp, queue1[20], queue2[20], temp1=0, temp2=0;

 // float avg;


  cout<<"Enter the no of request";

  cin>>q_size;
```

```
cout<<"Enter initial head position";
cin>>head;

cout<<"Enter the request:";

for(i=0; i<q_size; i++){
  cin>>temp;
  //queue1 - elems greater than head
  if(temp >= head){
      queue1[temp1] = temp;
      temp1++;
  } else {
      queue2[temp2] = temp;
      temp2++;
  }
}

//sort queue1 - increasing order
for(i=0; i<temp1-1; i++){
 for(j=i+1; j<temp1; j++){
     if(queue1[i] > queue1[j]){
       temp = queue1[i];
       queue1[i] = queue1[j];
       queue1[j] = temp;
     }
  }
}
```

```
//sort queue2 - decreasing order
for(i=0; i<temp2-1; i++){
  for(j=i+1; j<temp2; j++){
    if(queue2[i] < queue2[j]){
        temp = queue2[i];
        queue2[i] = queue2[j];
        queue2[j] = temp;
    }
  }
}

if(abs(head-LOW) >= abs(head-HIGH)){

    for(i=1,j=0; j<temp1; i++,j++){
        queue[i] = queue1[j];
    }

    for(i=temp1+1, j=0; j<temp2; i++, j++){
        queue[i] = queue2[j];
    }

} else {

    for(i=1,j=0; j<temp2; i++,j++){
        queue[i] = queue2[j];
    }

    for(i=temp2+1, j=0; j<temp1; i++, j++){
        queue[i] = queue1[j];
```

```
      }

  }

  queue[0] = head;

  for(j=0; j<q_size; j++){
      diff=abs(queue[j+1] - queue[j]);
        seek += diff;
        cout<<"Disk head moves from "<<queue[j]<<" to "<<queue[j+1]<<" with seek "<<diff<<endl;

  }

  cout<<"Total seek time is "<<seek;
 // avg = seek/(float)q_size;
 // printf("Average seek time is %f\n", avg);

  return 0;
}
```

## C-LOOK

Code:

```
#include <iostream>

#include <stdlib.h>

#define LOW 0

#define HIGH 199

using namespace std;
```

```
int main(){
  int queue[20], head, q_size, i,j, seek=0, diff, max, min, range, temp, queue1[20], queue2[20], temp1=0,
temp2=0;
  float avg;

 cout<<"Enter the no of request";
  cin>>q_size;

  cout<<"Enter initial head position";
  cin>>head;

  cout<<"Enter the request:";

  for(i=0; i<q_size; i++){
    cin>>temp;
    //queue1 - elems greater than head
    if(temp >= head){
      queue1[temp1] = temp;
      temp1++;
    } else {
      queue2[temp2] = temp;
      temp2++;
    }
  }


  //sort queue1 - increasing order
  for(i=0; i<temp1-1; i++){
```

```
  for(j=i+1; j<temp1; j++){
    if(queue1[i] > queue1[j]){
      temp = queue1[i];
      queue1[i] = queue1[j];
      queue1[j] = temp;
    }
  }
}

//sort queue2
for(i=0; i<temp2-1; i++){
  for(j=i+1; j<temp2; j++){
    if(queue2[i] > queue2[j]){
      temp = queue2[i];
      queue2[i] = queue2[j];
      queue2[j] = temp;
    }
  }
}

if(abs(head-LOW) <= abs(head-HIGH)){

  for(i=1,j=temp2-1; j>=0; i++,j--){
      queue[i] = queue2[j];
  }

  queue[i] = LOW;
  queue[i+1] = HIGH;
  for(i=temp2+3,j=temp1-1; j>=0; i++,j--){
      queue[i] = queue1[j];
  }

} else {

  for(i=1,j=0; j<temp1; i++,j++){
      queue[i] = queue1[j];
  }

  queue[i] = HIGH;
  queue[i+1] = LOW;

  for(i=temp1+3,j=0; j<temp2; i++,j++){
      queue[i] = queue2[j];
  }



}
queue[0] = head;

for(j=0; j<q_size; j++){
  diff=abs(queue[j+1] - queue[j]);
  seek += diff;
      cout<<"Disk head moves from "<<queue[j]<<" to "<<queue[j+1]<<" with seek "<<diff<<endl;

}
//range = max - min;
//printf("Range is %d", range);
```

```
  //seek =  seek - (max - min);


  cout<<"Total seek time is "<<seek;

  avg = seek/(float)q_size;

  printf("Average seek time is %f\n", avg);


  return 0;

}
```

## Banker's Algorithm

Code:

```
#include<iostream>

using namespace std;

int main(){

	int p[10],allocation[10][10],max[10][10],available[10],n1,n2,i,j,need[10][10],safe,seq,order[10];

	cout<<"Enter the no. of process(max 10)"<<endl;

	cin>>n1;

	cout<<"Enter the no. of resources(max 10)"<<endl;

	cin>>n2;

	for(i=0;i<n1;i++){

		//cout<<"Enter the value for process "<<i+1<<endl;

		cout<<"Enter the allocation for process "<<i+1<<endl;


		for(j=0;j<n2;j++){

			cin>>allocation[i][j];

		}

		cout<<"Enter the maximum required for process"<<i+1<<endl;

		for(j=0;j<n2;j++){

			cin>>max[i][j];
```

```
        }
}
cout<<"Enter the available resources"<<endl;
for(j=0;j<n2;j++){
                cin>>available[j];
}
for(i=0;i<n1;i++){
        cout<<"Resources needed for process "<<i+1<<endl;
        for(j=0;j<n2;j++){
                need[i][j]=max[i][j]-allocation[i][j];
                cout<<need[i][j]<<" ";
        }
        cout<<"\n";
        order[i]=0;
}
do{
for(i=0;i<n1;i++){
        safe=1;
        if(order[i]==0){

        cout<<"For Process "<<i+1<<endl;
        for(j=0;j<n2;j++){
                if(need[i][j]>available[j]){
                        safe=0;
                }
        }
        if(safe==1){
                cout<<"Process "<<i+1<<" is in safe state"<<endl;
                for(j=0;j<n2;j++){
                    available[j]=available[j]+allocation[i][j];
                }
                seq++;
                order[i]=seq;
            }
            else{
                cout<<"Process "<<i+1<<" is not in safe state"<<endl;

            }
        }
        }
}while(seq<n1);
cout<<"Sequence of process:";
for(i=0;i<n1;i++){
        for(j=0;j<n1;j++){
            if(order[j]==i+1)
            cout<<"process "<<j+1<<">>";
        }
}
}
```